International Journal of Technical Innovation in Modern Engineering & Science (IJTIMES)
Impact Factor: 3.45 (SJIF-2015), e-ISSN: 2455-2584
Volume 3, Issue 3, March-2017

ANALYSIS OF RADIX 2^4 SDF PIPELINE FFT ARCHITECTURE

[1]J.Eindhumathy, [2]M.Anthuvan Lydia, [3]V.Ramya

Assistant Professor,
Dept. of Electronics and Communication , Saranathan College of Engineering,Trichy-620002,India

Abstract— This paper describes the implementation of a R2^4 SDF Pipeline FFT Architecture using hardware description language Verilog HDL simulated up to 1 MHz for transformation length 16-point. A hardware oriented radix-2^4 algorithm is derived by integrating a twiddle factor in the folded transformation technique. The single-path delay feedback architecture is used to exploit the spatial regularity in signal flow graph of an algorithm [7]. For length-N DFT computation, the hardware requirement of proposed architecture is minimal on both dominant components: log$_4$N-1 complex multipliers and N-1 complex data memory. In this paper, we present a 16-point FFT module, which reduces the multiplicative complexity by using both real and complex constant multiplications. In order to reduce the hardware complexity pruning technique is used.

Keyword-FFT, Twiddle Factor, Butterfly Unit, multiplier, radix-2^4

I. INTRODUCTION

The Fast Fourier Transform (FFT) is one of vital algorithms in the field of digital signal processing. It is used to calculate the discrete Fourier transform (DFT) efficiently. In this context, pipelined hardware architectures are used because they offer high throughputs and low latencies suitable for real time, as well as a reasonably low area and power consumption as one of the efficient architecture. There are two main types of pipelined architectures: feedback (FB) and feed forward (FF). Feedback architectures are characterized by their feedback loops, i.e., the memories at the same stage [6]. Feedback architectures can be divided into Single-path Delay Feedback (SDF) which processes a continuous flow of one sample per clock cycle, and Multi-path Delay Feedback (MDF) or parallel feedback, which process several samples in parallel. On the other hand, feed forward architectures, also known as Multi-path Delay Commutator (MDC), do not have feedback loops and each stage passes the processed data to the next stage. These architectures can also process several samples in parallel. The Fast Fourier Transform (FFT) and its inverse (IFFT) are very imperative algorithms in signal processing, software-defined radio [9], and the most promising modulation technique (Orthogonal Frequency Division Multiplexing (OFDM)).Our FFT architecture is implemented in Radix-2 Single-path delay feedback (R2SDF) architecture with reduced memory.

Radix-2^2 FFT Algorithm:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot W_N^{nk}, \qquad 0 \le k < N \qquad (1)$$

Where $W_N = e^{\frac{-j2\pi}{N}}$

According to the decomposition method of [6] that done by substituting with

$$n = \langle \frac{N}{2} n_1 + \frac{N}{4} n_2 + n_3 \rangle_N$$
$$k = \langle k_1 + 2k_2 + 4k_3 \rangle_N$$

This yield

$$X(k_1 + 2k_2 + 4k_3) = \sum_{n_3=0}^{\frac{N}{4}-1} \left\{ H(k_1, k_2, n_3) \cdot W_N^{n_3(k_1+2k_2)} \right\} W_{\frac{N}{4}}^{n_3 k_3} \qquad (2)$$

The Discrete Fourier Transform (DFT) of N-point input *x(n)* is defined

With

$$H(k_1,k_2,n_3) = \left[x(n_3) + (-1)^{k_1} \cdot x(n_3 + \frac{N}{2})\right] + (-j)^{(k_1+2k_2)} \left[x(n_3 + \frac{N}{4}) + (-1)^{k_1} \cdot x(n_3 + \frac{3N}{4})\right] \quad (3)$$

After this simplification we have a set of four DFTs of length N/4.[8] Each term in equation (3) represents a Radix-2 butterfly (BFI), while the whole equation represents Radix-2 butterfly (BFII) with trivial multiplication by(-*j*). Notice the order of input and output; the inputs are in normal order whereas the outputs are in permuted (digit-reversed) order. The pentagon's between BFI and BFII represent the trivial multiplication by –*j*. After these two butterflies, full twiddle factor multipliers (TFM) are required to compute the multiplication by the twiddle factor $W^{n3(k1+2k2)}$

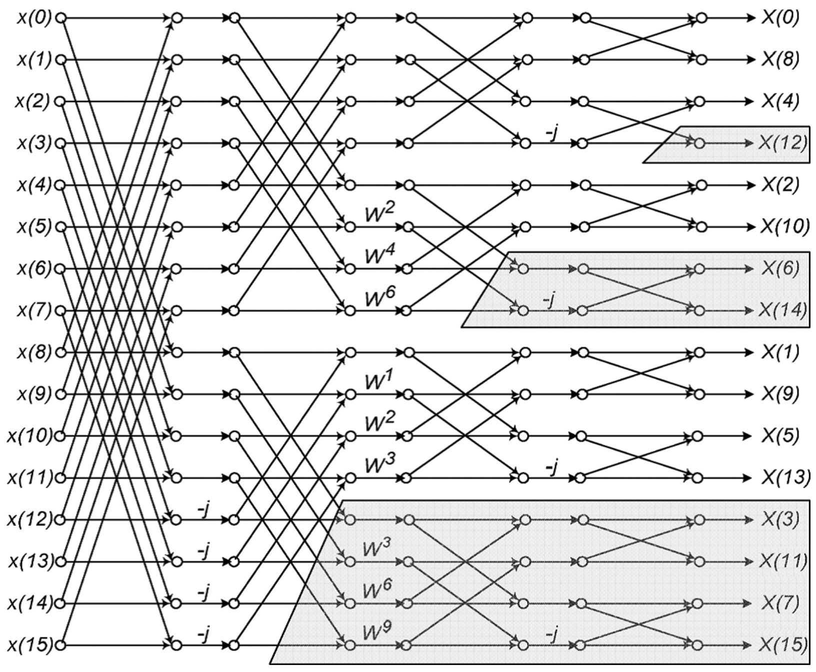


Fig 1: Flow graph of the 16-point radix-2^2 DIF FFT.

II. Butterfly Structure:

(i) BF1 Structure:

This is one of the basic blocks used in the IFFT/FFT implementation. The structure is as given below. It's implemented in accordance with the signal flow graph. The structure is always associated with a specific number of shift registers as specified in the overall structure. If there are N such registers, then for the first N/2 cycles, the inputs which keep arriving serially, are stored in the shift register. Then from the N/2+1 the cycle onwards, the butterfly begins its computation. This action is controlled by the 's' signal. When the's' signal goes high, the butterfly is enabled as seen in the diagram. When it's off, the butterfly is essentially inactive because the multiplexer simply passes whatever input it receives. This is stored in the corresponding shift register. Here, xr(n) and xi(n) represent the set of inputs which is given to the butterfly structure and xr(n+N/2) and xi(n+N/2) represent the set of inputs which come from the shift register.

Shift Registers:

Shift registers are implemented as shown in the block diagram. They store the incoming values till they are filled and then they are taken out one by one to be used in the corresponding butterfly structure.

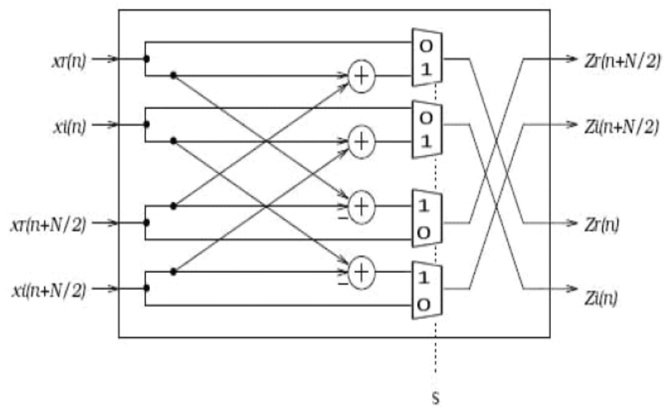


Fig 2- BF1 Structure

(ii) BF2 Structure

This is the other basic block used in the FFT/IFFT implementation. It's almost similar to the BF1 block. The major difference is it allows for the pre multiplication by (-i) and then performing the basic butterfly operation. This is in accordance with the signal flow graph. The pre multiplication by (-i) is based on the following algorithm:

$$[a + bi] X [-i] = b - ai$$

Basically, the real and imaginary parts of the input complex number are swapped and the imaginary part of the output is negated. Referring to the butterfly diagram, it's found that this is to be done in the second half of s being high. This is carried out in the form of signal where a is added with s controls the swapping. If this control is low, the value is passed as such. After this, the corresponding outputs are passed inside the structure and then control is based on the 's' signal as before. Now, operation is similar to the BF1 structure.

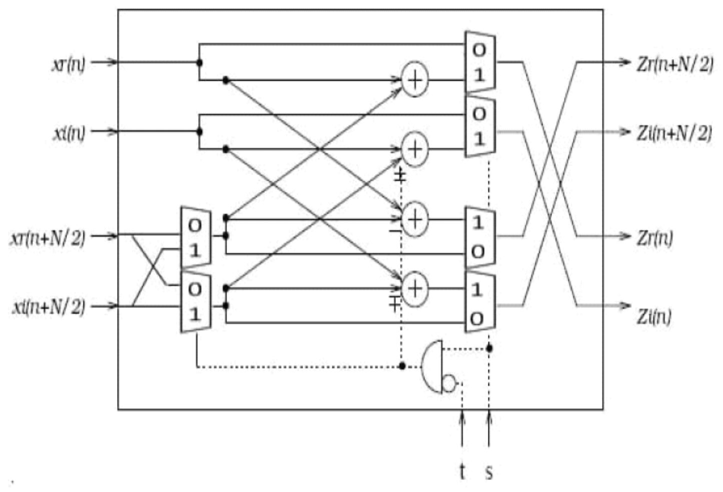


Fig 3– BF2 Structure

III.RADIX-2^2 SDF FFT Architecture

SDF FFT architectures make use of delay-lines implemented using memory and shift registers to reorder data at each butterfly stage. Delay-lines of length 2^m are required for all m from 0 to $\log_2(N)- 1$ where N is the number of FFT points the SDF FFT processor is capable of computing[2]. This requirement is due to the data shuffling intrinsic to the decimate-in-time (DIT) and decimate-in-frequency (DIF) algorithms. The Radix-2^2 SDF architecture is a hybrid of Radix-2 SDF and Radix-4 SDF designs. The simplicity of the Radix-2 two-point butterfly structure is maintained while only needing $\log_4 (N)$-2 twiddle multiplies as is the case in Radix-4 architectures. This flexibility is achieved by using a second type of butterfly structure that performs multiplications through sign inversion and real-imaginary sample swapping. This simplification eliminates half of the complex multipliers required for Radix-2 SDF implementations.

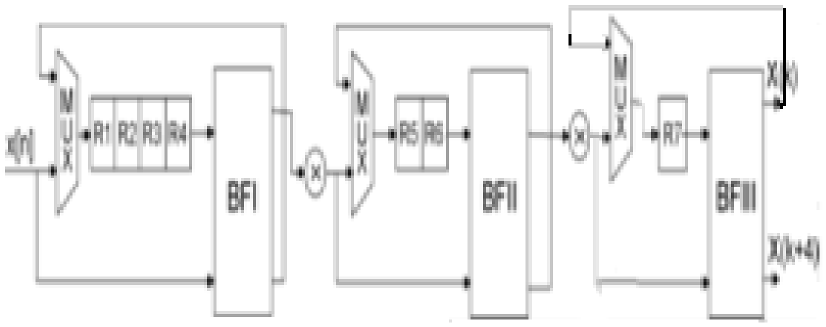


Fig 4: 8 point RADIX 2^2 SDF Architecture

FFT design used in this paper is SDF (single path delay feedback) architecture using folded transformation. Here many butterflies in the same column can be mapped to 1 butterfly unit.The main objective is for achieving high throughput rates , all the processors share the rotation memory in order to reduce hardware and latency.[2] In this 3 operations are to be performed. (i) **Selecting the mux**. For the 1st half samples, control signal must be zero. In that time, input is entered into the mux and it is stored in the register .If it starts subtraction operation control(ctrl) signal must be set to 1.Then only corresponding subtracted value is stored in the register element.(ii) corresponding mux o/p is entered into the register. Then Right **shift operation** takes place here (iii) For the 1st stage , subtraction operation is performed on the 9th clock onwards. In this time, ctrl signal must be set to 1.So r8 content will be added and subtracted with x(8) input and sum o/p is stored in the separate variable. Difference o/p is feedback to one of the input of mux which is again fed into the shift register. Later 1st stage sum output is fed as input to 2nd stage. Same process will be continued for every stage. If the stages of FFT increase corresponding register elements used will be very less. Radix2^2 FFT processor achieves lowest execution time and highest operating frequency.

IV. RADIX-2^4 SDF FFT Architecture

Radix-2^4 SDF FFT architecture is simpler than that of the direct radix-16 FFT SDF structure. The multiplier cost of the proposed FFT architecture is less than that of the previous FFT structures in Radix 2^2 FFT applications. The throughput of the proposed FFT processor is one sample per clock..

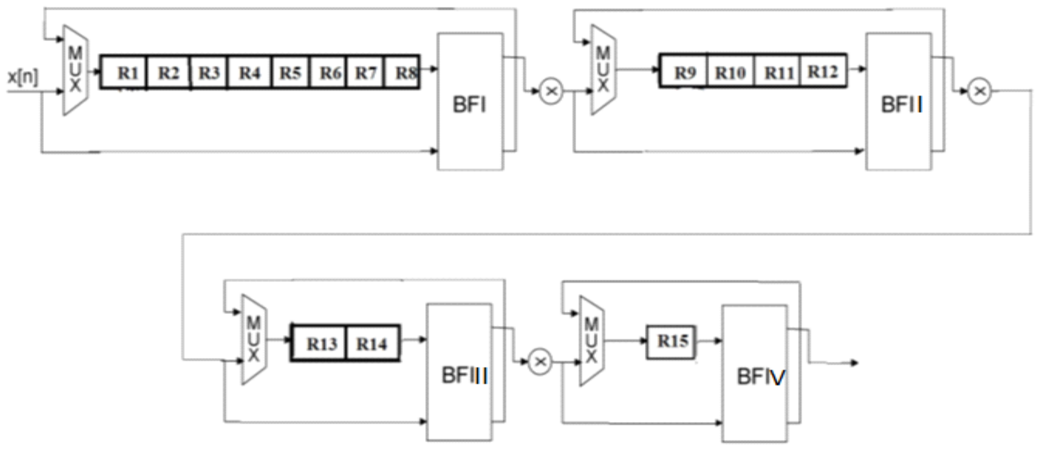


Fig5: 16 point Radix - 2^4 SDF Architecture

This architecture is similar to the Radix 2^2.The difference between radix 2^2 and radix 2^4 is twiddle factor multiplication only.

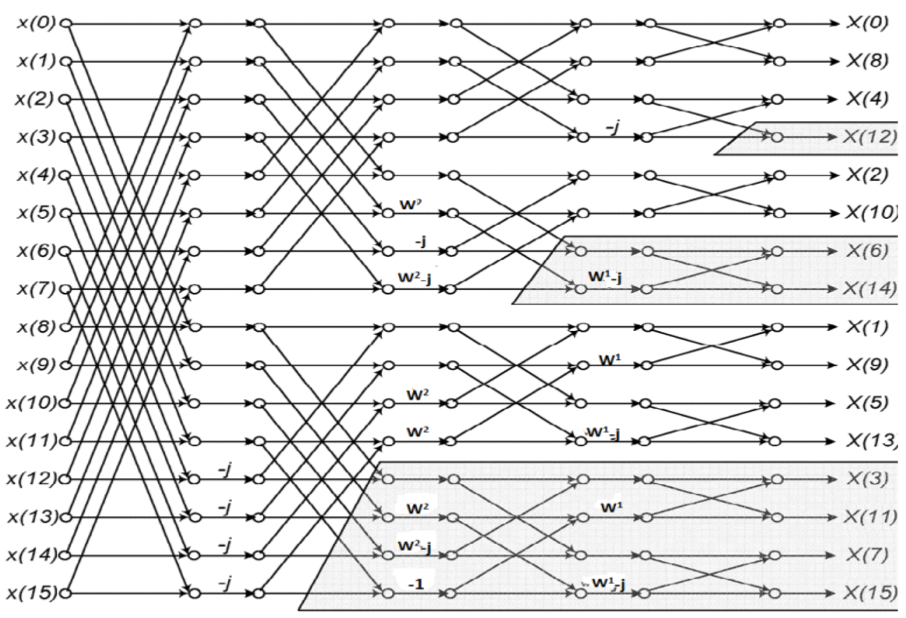


Fig 6: Flow graph of the 16-point radix-2^4 DIF FFT.

PRUNING:
To increase the efficiency of the FFT technique several pruning and different other techniques have been proposed by many researchers [5]. In this paper, a new pruning technique i.e. IZ-TFFTP by simple modification and some changes in the conventional flowchart of FFT and it also includes pipeline techniques to reduce the total execution time.
Zero tracing - as in wide band communication system a large portion of frequency channel may be unoccupied by the licensed user, so no. of zero valued inputs are much greater than the non-zero valued inputs in a FFT/IFFT operation at the transceiver. Then this algorithm [1] will give best response in terms of reduced execution time by skip the operation as they are dual node to each other reducing the number of complex computation required for twiddle factor calculation [9]. Pruning has a strong searching condition, which have a 2-D array for storing the input & output values after every iteration of butterfly calculation. In a input searching result whenever it found "zero" at any input, simply omit that calculation.

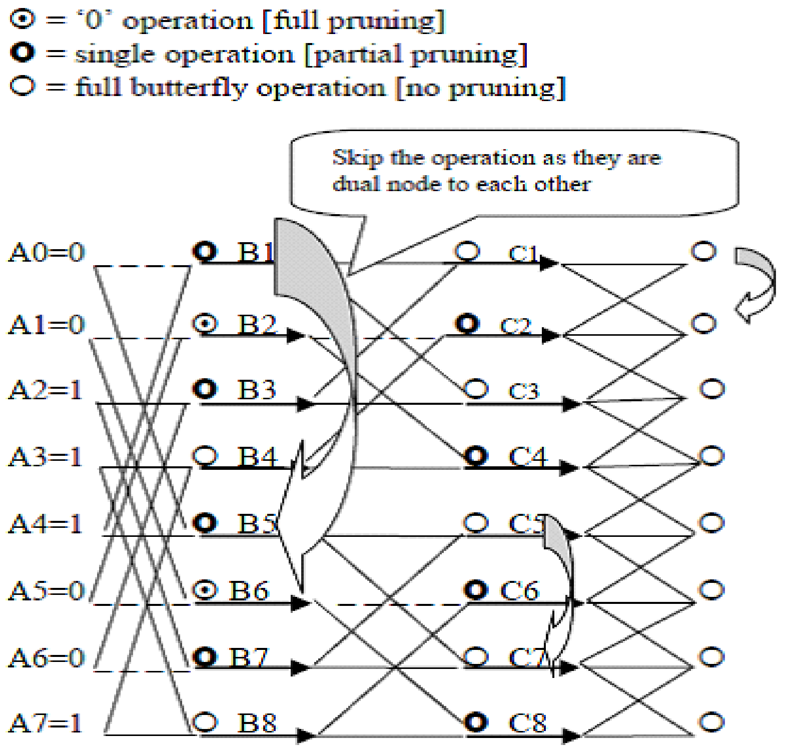
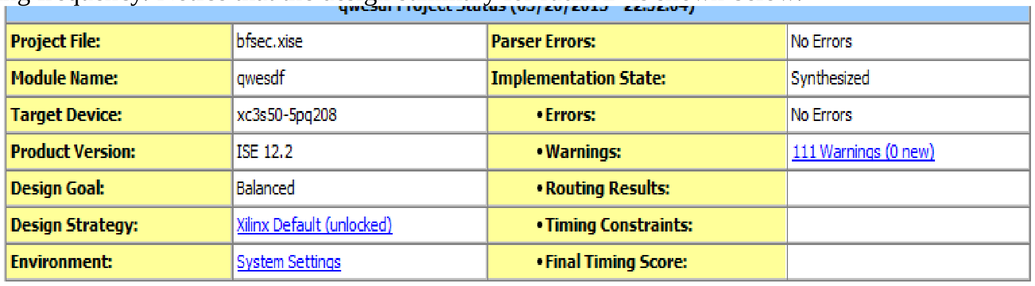

⊙ = '0' operation [full pruning]
● = single operation [partial pruning]
○ = full butterfly operation [no pruning]
Skip the operation as they are dual node to each other
A0=0
A1=0
A2=1
A3=1
A4=1
A5=0
A6=0
A7=1
B1
B2
B3
B4
B5
B6
B7
B8
C1
C2
C3
C4
C5
C6
C7
C8

Fig7: point DIF-FFT flow graph

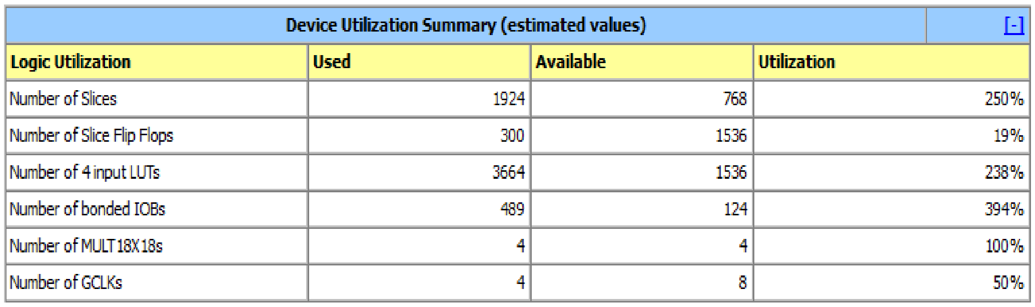
V. Implementation Results

The results of the timing analysis using the ModelSim Xilinx Edition (MXE III) simulator indicate a maximum operating frequency of 465 MHz(Radix 2^2) and 30.317MHz(Radix 2^4). This provides a minimum execution time of a 16 complex data points is 32.985ns (Radix 2^4).

Table 1 shows a comparison with some of FFT processors. Our Radix-2^4 processor achieves lowest execution time and highest operating frequency. Notice that the design summary for radix 2^4 is shown below:



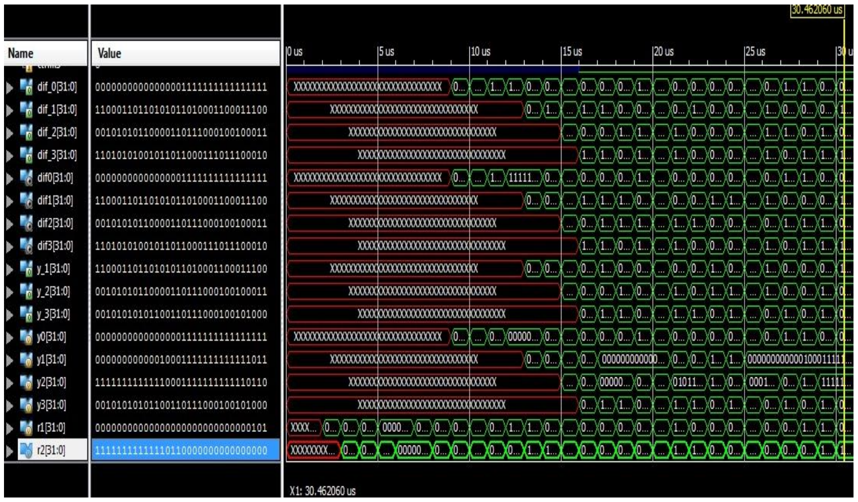
Project File:	bfsec.xise	Parser Errors:	No Errors
Module Name:	qwesdf	Implementation State:	Synthesized
Target Device:	xc3s50-5pq208	• Errors:	No Errors
Product Version:	ISE 12.2	• Warnings:	111 Warnings (0 new)
Design Goal:	Balanced	• Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	
Environment:	System Settings	• Final Timing Score:	



Device Utilization Summary (estimated values) [-]

Logic Utilization	Used	Available	Utilization
Number of Slices	1924	768	250%
Number of Slice Flip Flops	300	1536	19%
Number of 4 input LUTs	3664	1536	238%
Number of bonded IOBs	489	124	394%
Number of MULT18X18s	4	4	100%
Number of GCLKs	4	8	50%

The result of radix 2^4 SDF is shown below :



```
Name        Value
dif_0[31:0] 00000000000000001111111111111111
dif_1[31:0] 11000110110101011010001100011100
dif_2[31:0] 00101010110000110111000100100011
dif_3[31:0] 11010101001011011000111011100010
dif0[31:0]  00000000000000001111111111111111
dif1[31:0]  11000110110101011010001100011100
dif2[31:0]  00101010110000110111000100100011
dif3[31:0]  11010101001011011000111011100010
y_1[31:0]   11000110110101011010001100011100
y_2[31:0]   00101010110000110111000100100011
y_3[31:0]   00101010101100110111000100101000
y0[31:0]    00000000000000001111111111111111
y1[31:0]    00000000000010001111111111111011
y2[31:0]    11111111111110001111111111110110
y3[31:0]    00101010101100110111000100101000
r1[31:0]    00000000000000000000000000000101
r2[31:0]    11111111111110110000000000000000
X1: 30.462060 us
```

VI. CONCLUSION

This paper has proposed a set of extensions that can be used to apply parallelism to the Radix-2^4 SDF pipeline. The proposed methods are flexible and allow for N-point FFT and IFFT computations. Additionally, both the DIF and the DIT algorithms are supported. Although the stated extensions apply specifically to the Radix-2^4SDF algorithm, similar techniques could be used for all pipelined SDF FFT implementations. The proposed extensions impose no restrictions on the overall throughput of the FFT circuit given adequate resource availability.There is a lot of scope for future development in this area. This improves area efficiency since three modulators can be replaced with one at the transmitter and the same at receiver. Reconfigurability can be implemented in the FFT and IFFT blocks also so that different point FFTs suited to different modulation schemes can be implemented which enhances the efficiency of these modulation schemes.

References

1. Markel, J. D. (1971) FFT Pruning. IEEE Trans. on Audio and Electro acoustics, 19, 305-311.
2. Manohar Ayinala," Pipelined Parallel FFT Architectures via Folding Transformation", IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL .20,JUNE
3. P. N. Whatmough, M. R. Perrett, S. Isam, and I. Darwazeh, "VLSI architecture for a reconfigurable spectrally efficient FDM baseband transmitter," in Proc. IEEE Int. Symp. Circuits Syst., May 2011, pp. 1688–1691.
4. Ahmadinia, A., B. Ahmad and T. Arslan **(**2007) System Level Modelling of Reconfigurable FFT Architecture for System-on-Chip Design. Proc. 2nd NASA/ESA Conf. Adaptive Hardware and Systems, 169-175
5. R. G. Alves, P. L. Osorio, and M. N. S. Swamy, "General FFT pruning algorithm," in Proc. 43rd IEEE Midwest Symp. Circuits Syst., (2000),vol. 3, pp. 1192–1195.
6. Mario Garrido, Member, IEEE, J. Grajal, M. A. Sánchez, and Osear Gustafsson, SéniorMember, IEEE ," Pipelined Radix-2 Feedforward FFT Architectures"
7. AHMED SAEED, M. ELBABLY,G. ABDELFADEEL and M. I. ELADAWY
8. "FPGA implementation of Radix-22 Pipelined FFT Processor ", Future University in Egypt, Helwan University,Helwan City, Cairo, EGYPT .